B.Sc. H Computer Science

Semester VI

# Data Mining

Lab Record

Name: shubham singh fartyal

Roll. No. – 19/78054

Table of Contents

**S.No. Program Title Page**

1. Question 1 3 2. Question 2 7

1. Question 3 13
2. Question 4 17
3. Question 5 19
4. Question 6 31

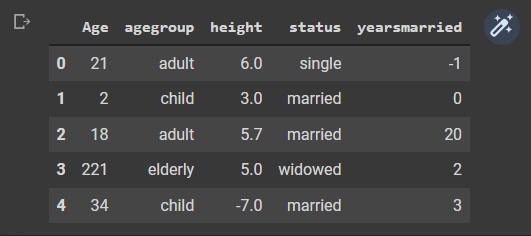
**Q1. Create a file “people.txt” with the following**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Age** | **agegroup** | **height** | **status** | **Years married** |
| 21 | adult | 6.0 | single | -1 |
| 2 | child | 3 | married | 0 |
| 18 | adult | 5.7 | married | 20 |
| 221 | elderly | 5 | widowed | 2 |
| 34 | child | -7 | married | 3 |

* 1. **i ) Read the data from the file “people.txt”.**
  2. Create a ruleset E that contain rules to check for the following conditions:
  3. The age should be in the range 0-150.
  4. The age should be greater than yearsmarried.
  5. The status should be married or single or widowed.
  6. If age is less than 18 the agegroup should be child, if age is between 18 and 65 the agegroup should be adult, if age is more than 65 the agegroup should be elderly. iii) Check whether ruleset E is violated by the data in the file people.txt.

iv) Summarize the results obtained in part (iii) v) Visualize the results obtained in part (iii)

|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt import ruleset    df = pd.read\_csv("people.txt",delimiter=' ')  df |



**ii) Create a ruleset E that contain rules to check for the following conditions:**

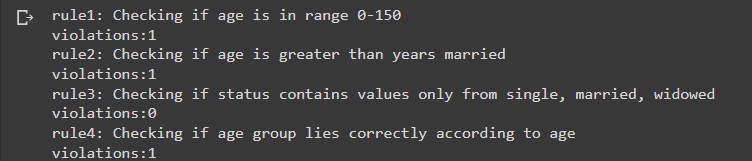
1. **The age should be in the range 0-150.**
2. **The age should be greater than years married.**
3. **The status should be married or single or widowed.**
4. **If age is less than 18 the age group should be child, if age is between 18 and 65 the age group should be adult, if age is more than 65 the age group should be elderly.**

Ruleset.py:

|  |
| --- |
| def age\_check(df):  errors = df["Age"][(df["Age"]>150)|(df["Age"]<0)].shape[0] return errors, "Checking if age is in range 0-150"  def age\_check2(df): n = df.shape[0] errors = 0 for i in range(n): if df["Age"][i]<df["yearsmarried"][i]:  errors+=1 return errors, "Checking if age is greater than years married"  def status\_check(df): errors = df.shape[0]df[df["status"].isin(['single','married','widowed'])].shape[0] return errors, "Checking if status contains values only from single, married, widowed"  def agegroup\_check(df):  n = df.shape[0] errors = 0 for i in range(n): if df["Age"][i]<18 and df["agegroup"][i]!="child":  errors+=1 elif df["Age"][i]>=18 and df["Age"][i]<65 and df["agegroup"][i]!="adult":  errors+=1 elif df["Age"][i]>=65 and df["agegroup"][i]!="elderly":  errors+=1 return errors, "Checking if age group lies correctly according to age" |

|  |
| --- |
| rules = [] rules.append(ruleset.age\_check) rules.append(ruleset.age\_check2) rules.append(ruleset.status\_check) rules.append(ruleset.agegroup\_check)  violations = [] rule = list(map(lambda x: "rule"+str(x), range(1,len(rules)+1))) for i in range(len(rules)):  violation,desc = rules[i](df) print(f"{rule[i]}: {desc}\nviolations:{violation}") violations.append(violation) |

**Output**

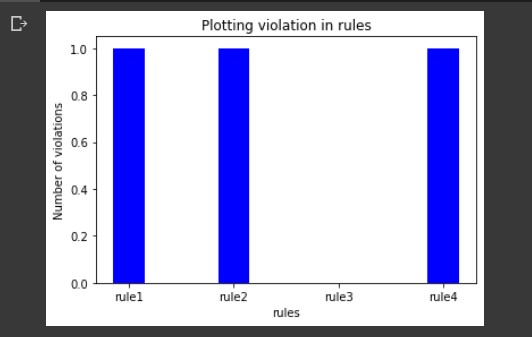


**Q1.**

**v) Visualize the results obtained in part (iii)**

fig = plt.figure() ax = fig.add\_subplot() ax.bar(rule,violations,0.3,color = 'blue') plt.ylabel("Number of violations") plt.xlabel("rules") plt.title("Plotting violation in rules") plt.show()

Output:



**Q2. Perform the following preprocessing tasks on the dirty\_iris datasetii.**

1. Calculate the number and percentage of observations that are complete.
2. Replace all the special values in data with NA.
3. Define these rules in a separate text file and read them.

(Use editfile function in R (package editrules). Use similar function in Python).

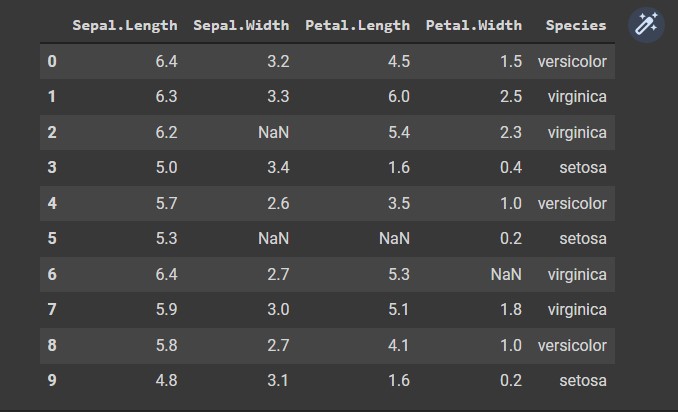
Print the resulting constraint object.

* Species should be one of the following values: setosa, versicolor or virginica.
* All measured numerical properties of an iris should be positive.
* The petal length of an iris is at least 2 times its petal width.
* The sepal length of an iris cannot exceed 30 cm.
* The sepals of an iris are longer than its petals.

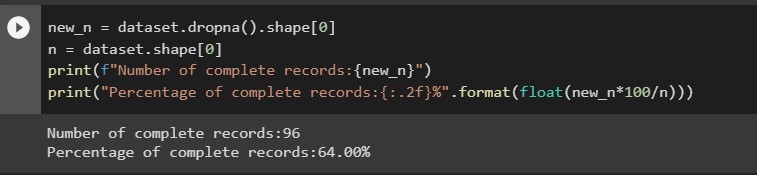
1. Determine how often each rule is broken (violatedEdits). Also summarize and plot the result.
2. Find outliers in sepal length using boxplot and boxplot.stats

|  |
| --- |
| import numpy as np import pandas as pd import matplotlib.pyplot as plt import ruleset  dataset = pd.read\_csv("dirty\_iris.csv")    dataset.head(10) |

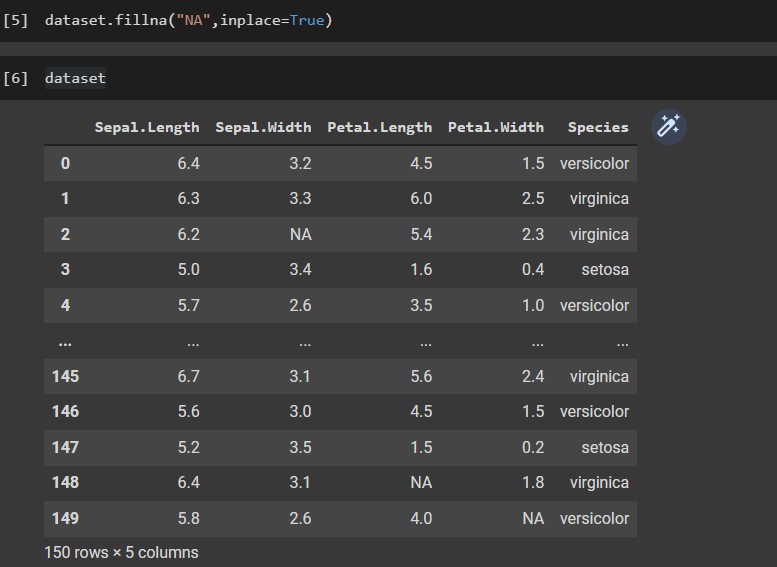
**Output**



1. Calculate the number and percentage of observations that are complete.



1. Replace all the special values in data with NA.



1. Define these rules in a separate text file and read them.

(Use editfile function in R (package editrules). Use similar function in Python).

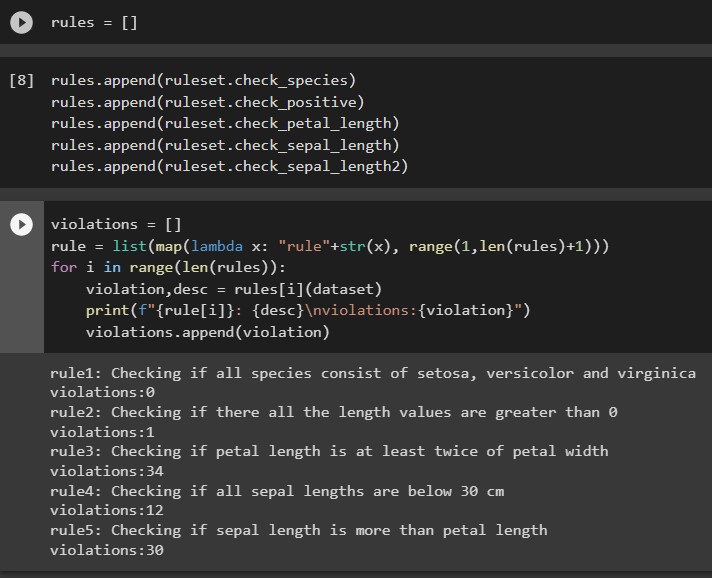
Print the resulting constraint object.

* Species should be one of the following values: setosa, versicolor or virginica.
* All measured numerical properties of an iris should be positive.
* The petal length of an iris is at least 2 times its petal width.
* The sepal length of an iris cannot exceed 30 cm.
* The sepals of an iris are longer than its petals.

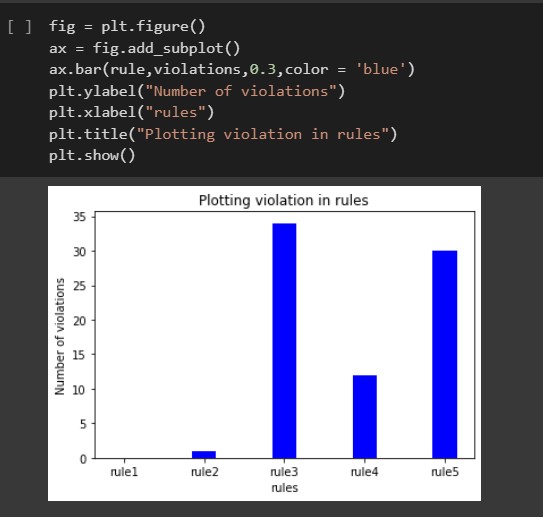
Ruleset:

|  |
| --- |
| def check\_species(dataset): n = dataset.shape[0] species = dataset["Species"][dataset["Species"]!='NA'] correct\_n = species[species.isin(["setosa","versicolor","virginica"])].shape[0] return n-correct\_n, "Checking if all species consist of setosa, versicolor and virginica"  def check\_positive(dataset):  violation\_n = 0 for i in dataset.values: for j in i: if isinstance(j,float): if j<0:  violation\_n+=1  return violation\_n, "Checking if there all the length values are greater than 0" def check\_petal\_length(dataset):  n = dataset.shape[0] correct\_n = 0 for i in range(n): if dataset["Petal.Width"][i]=='NA' or dataset["Petal.Length"][i]=='NA':  continue elif dataset["Petal.Width"][i]\*2>dataset["Petal.Length"][i]:  continue else:  correct\_n+=1  return n-correct\_n, "Checking if petal length is at least twice of petal width" |

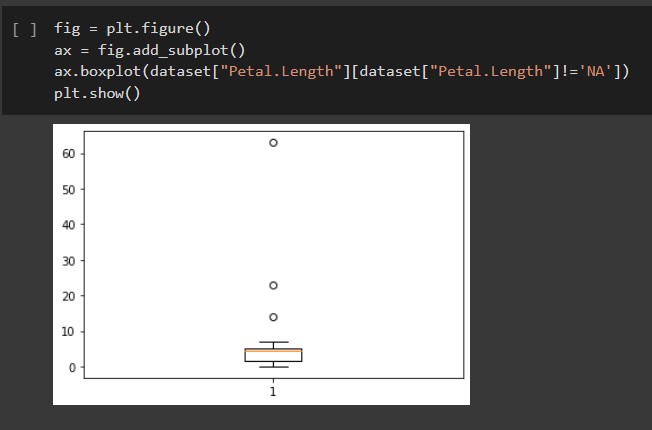
|  |
| --- |
| def check\_sepal\_length(dataset):  n = dataset.shape[0] sl = dataset["Sepal.Length"][dataset["Sepal.Length"]!='NA'] correct\_n = sl[sl<=30].shape[0] return n-correct\_n, "Checking if all sepal lengths are below 30 cm"  def check\_sepal\_length2(dataset):  n = dataset.shape[0] correct\_n = 0 for i in range(n): if dataset["Sepal.Length"][i]=='NA' or dataset["Petal.Length"][i]=='NA':  continue elif dataset["Petal.Length"][i]>=dataset["Sepal.Length"][i]:  continue else:  correct\_n+=1 return n-correct\_n, "Checking if sepal length is more than petal length" |



iv) Determine how often each rule is broken (violated Edits). Also summarize and plot the result.



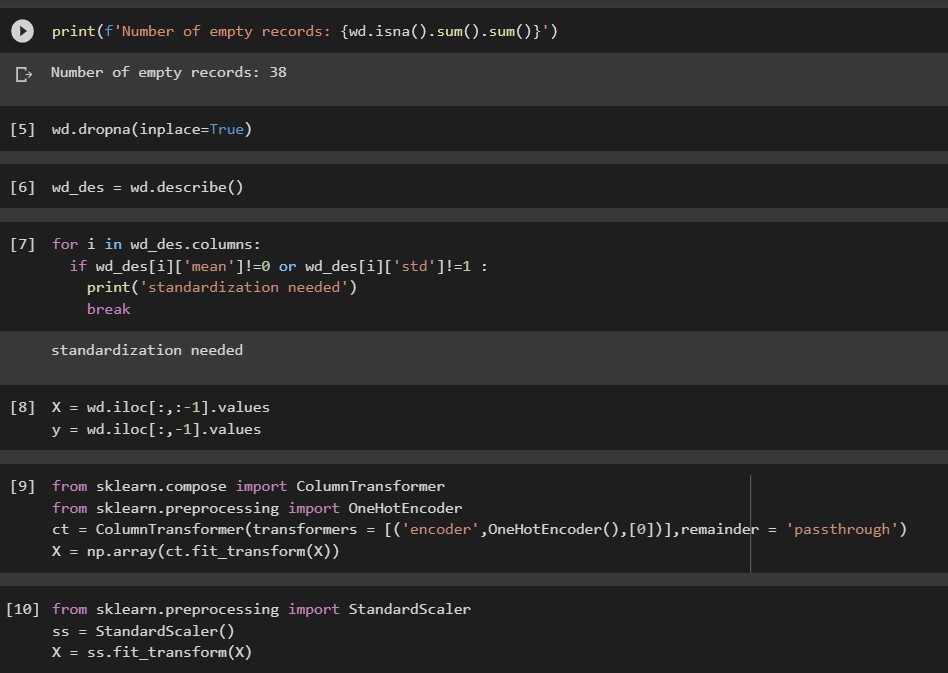
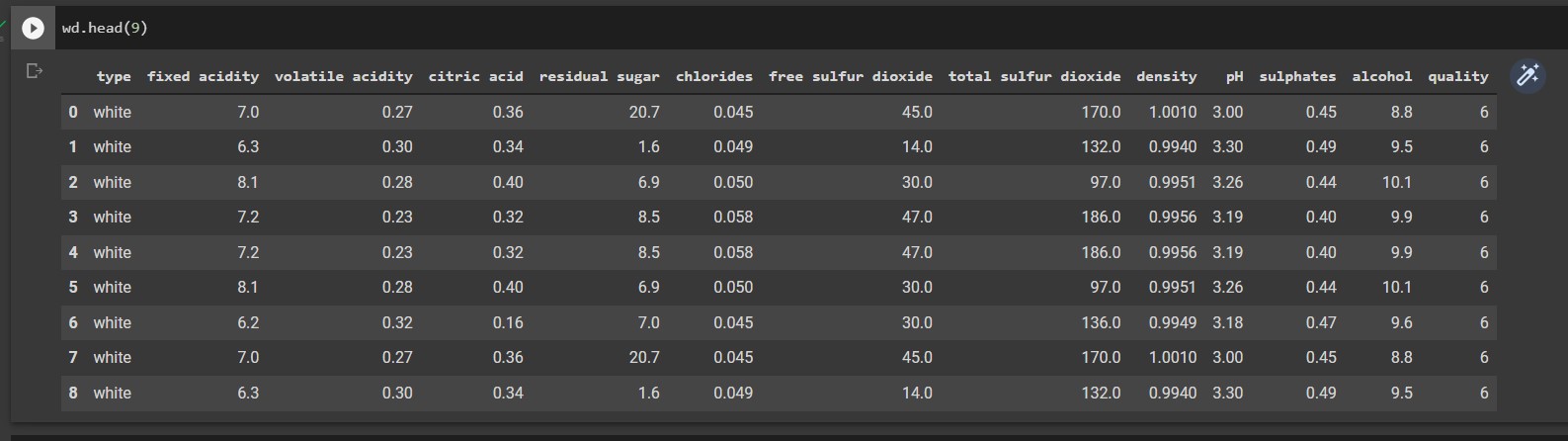
vi) Find outliers in sepal length using boxplot

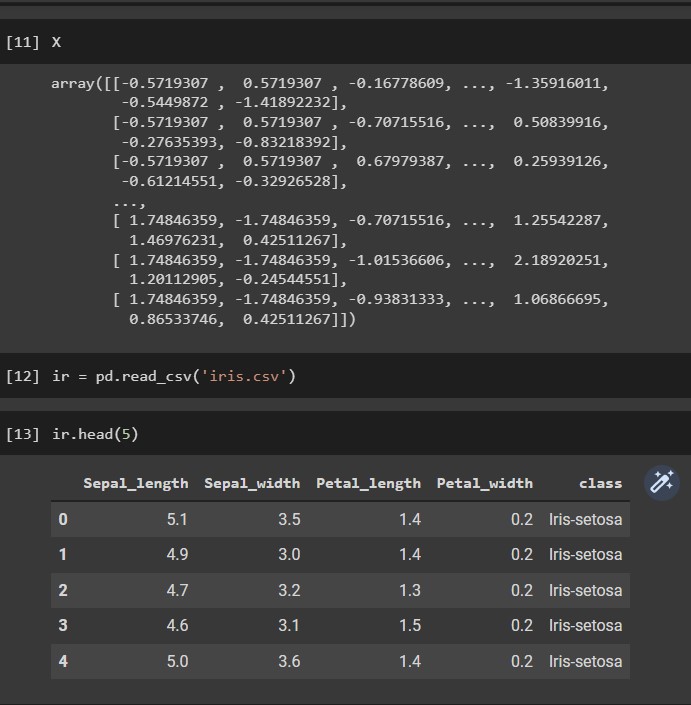


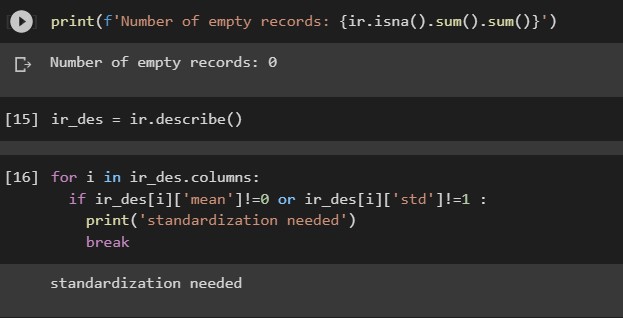
Q3. Load the data from wine dataset. Check whether all attributes are standardized or not (mean is 0 and standard deviation is 1). If not, standardize the attributes. Do the same with Iris dataset.

|  |  |
| --- | --- |
| import pandas as pd import numpy as np    # In[2]: wd = pd.read\_csv('winequalityN.csv')    # In[3]: wd.head(9)    # In[4]:  print(f'Number of empty records: {wd.isna().sum().sum()}')  # In[5]: wd.dropna(inplace=True)    # In[6]: wd\_des = wd.describe()    # In[7]: for i in wd\_des.columns: if wd\_des[i]['mean']!=0 or wd\_des[i]['std']!=1 :  print('standardization needed') break    # In[8]:  X = wd.iloc[:,:-1].values y = wd.iloc[:,-1].values    # In[9]:  from sklearn.compose import ColumnTransformer from sklearn.preprocessing import OneHotEncoder ct = ColumnTransformer(transformers =  [('encoder',OneHotEncoder(),[0])],remainder = 'passthrough')  X = np.array(ct.fit\_transform(X))    # In[10]: from sklearn.preprocessing import StandardScaler ss = StandardScaler() X = ss.fit\_transform(X) |  |
|  |  |
| # In[11]:  X    # In[12]: ir = pd.read\_csv('iris.csv')  # In[13]: ir.head(5)    # In[14]:  print(f'Number of empty records: {ir.isna().sum().sum()}')  # In[15]: ir\_des = ir.describe()    # In[16]: for i in ir\_des.columns: if ir\_des[i]['mean']!=0 or ir\_des[i]['std']!=1 :  print('standardization needed') break  # In[17]:  X2 = ir.iloc[:,:-1].values y2 = ir.iloc[:,-1].values    # In[18]: ss = StandardScaler()  X2 = ss.fit\_transform(X2)    # In[19]:  X2 |

Output:

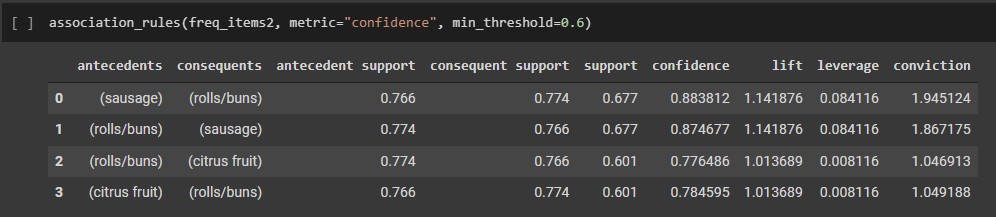
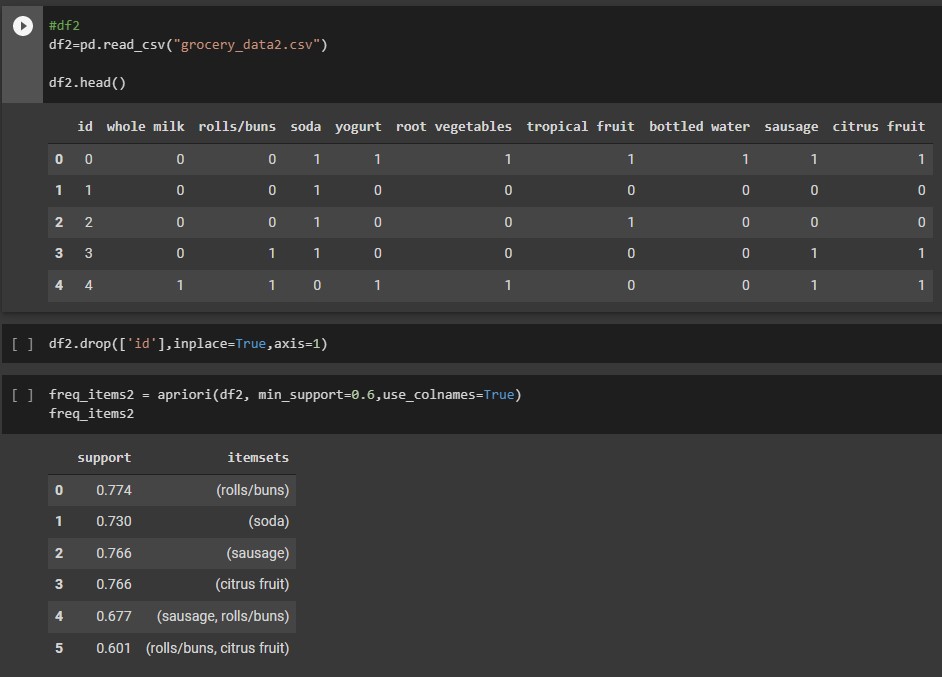
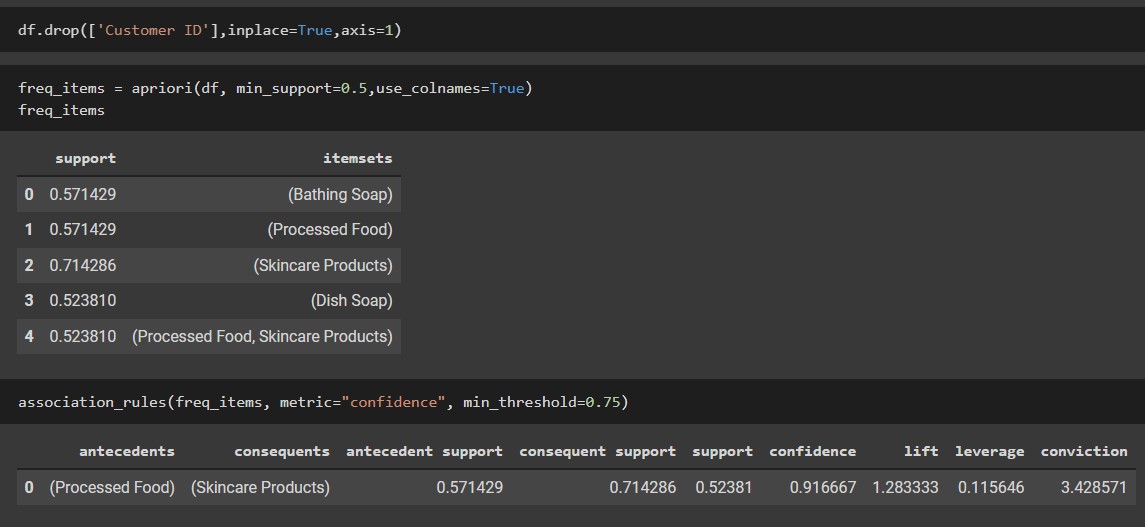
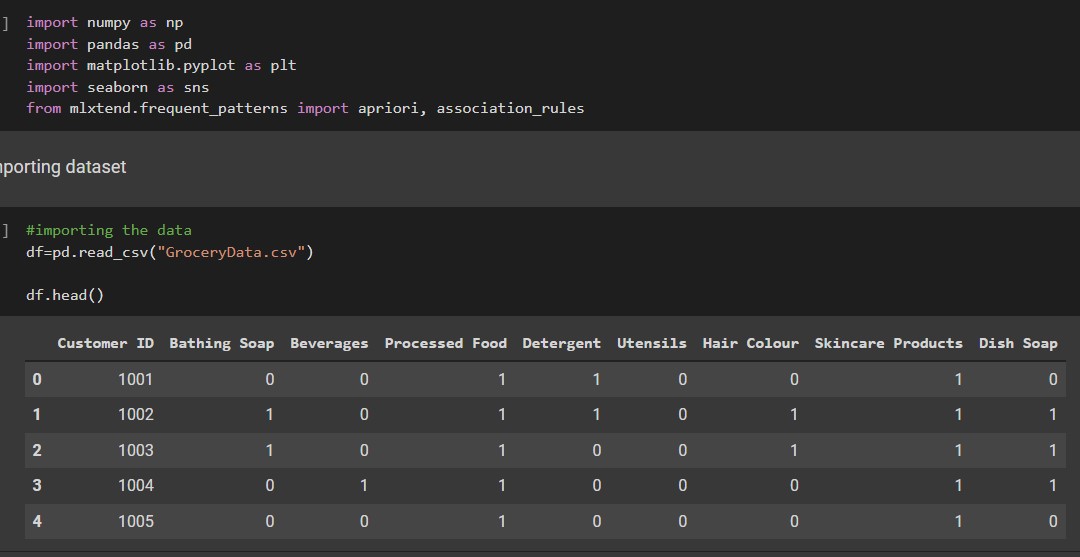






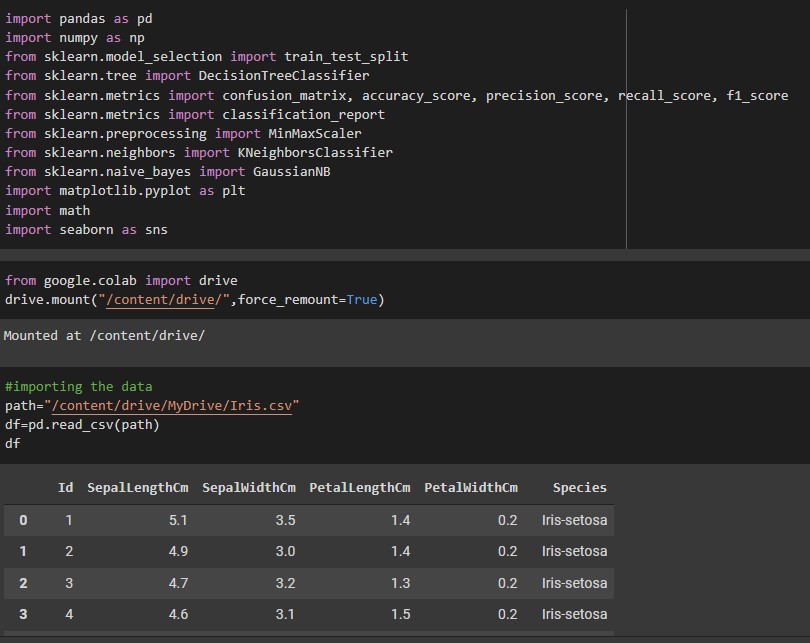
Q4. Run Apriori algorithm to find frequent item sets and association rules

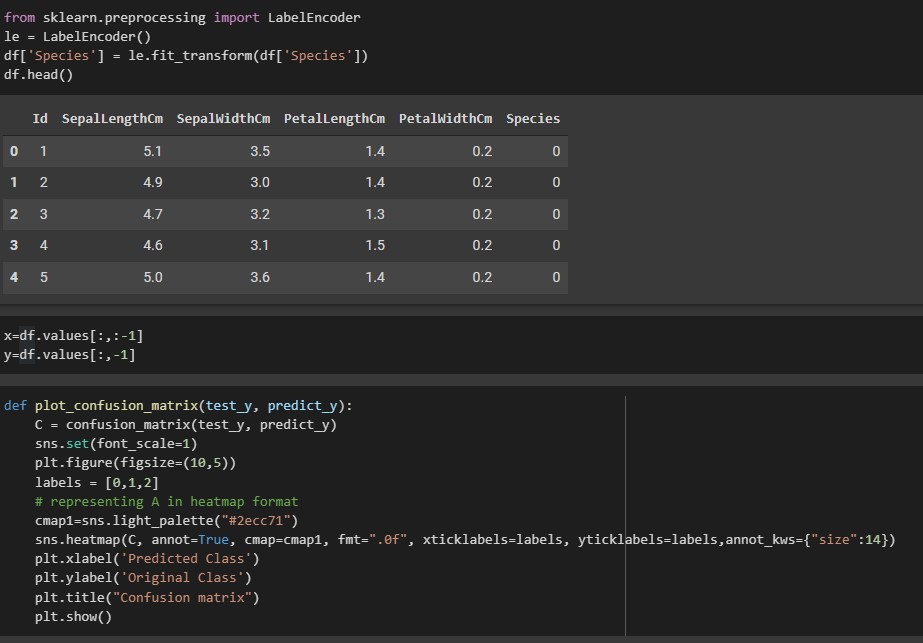
* 1. Use minimum support as 50% and minimum confidence as 75%
  2. Use minimum support as 60% and minimum confidence as 60 %

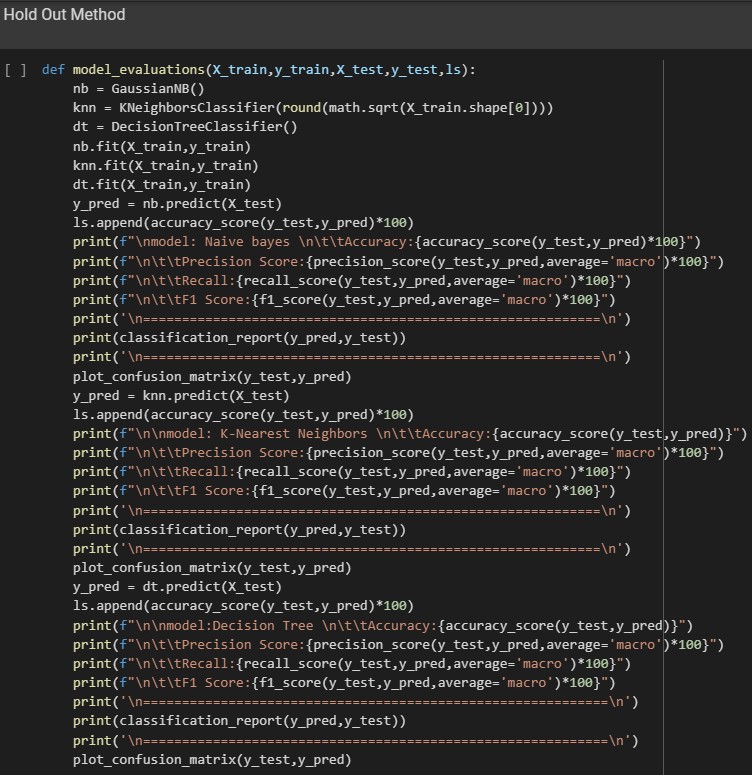


Q5. Use Naive bayes, K-nearest, and Decision tree classification algorithms and build classifiers. Divide the data set into training and test set. Compare the accuracy of the different classifiers under the following situations:

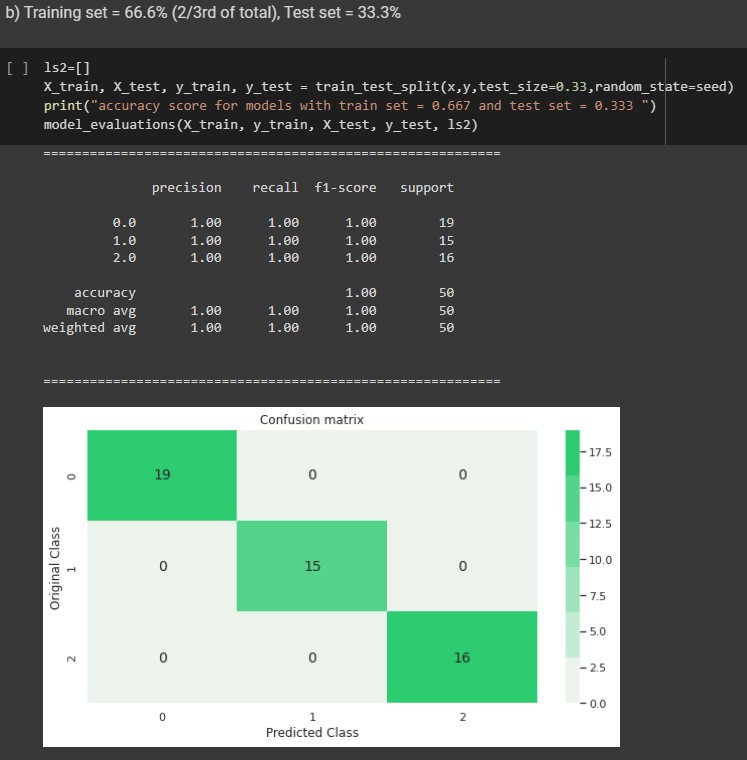
* 1. a) Training set = 75% Test set = 25% b) Training set = 66.6% (2/3rd of total), Test set = 33.3%
  2. Training set is chosen by i) hold out method ii) Random subsampling iii) CrossValidation. Compare the accuracy of the classifiers obtained.
  3. Data is scaled to standard format.

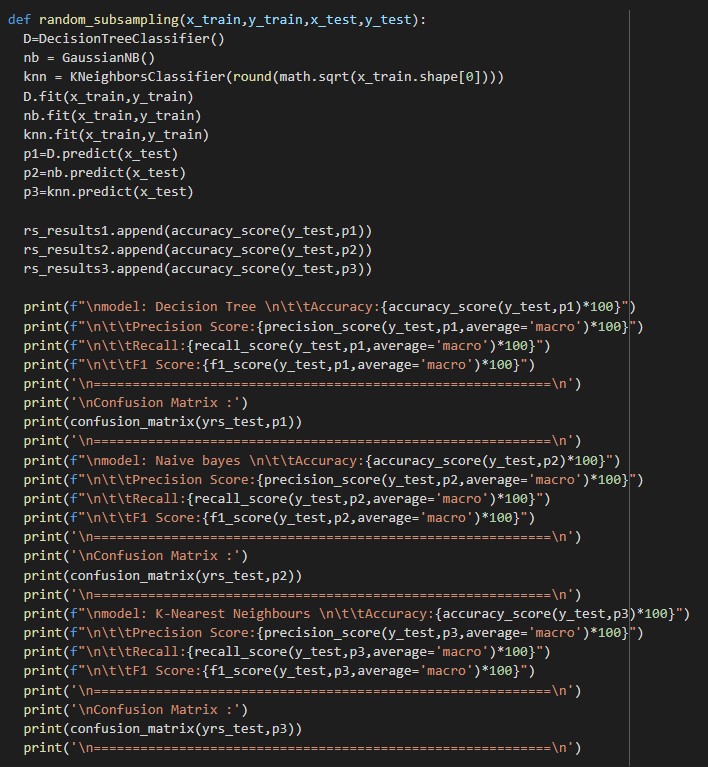




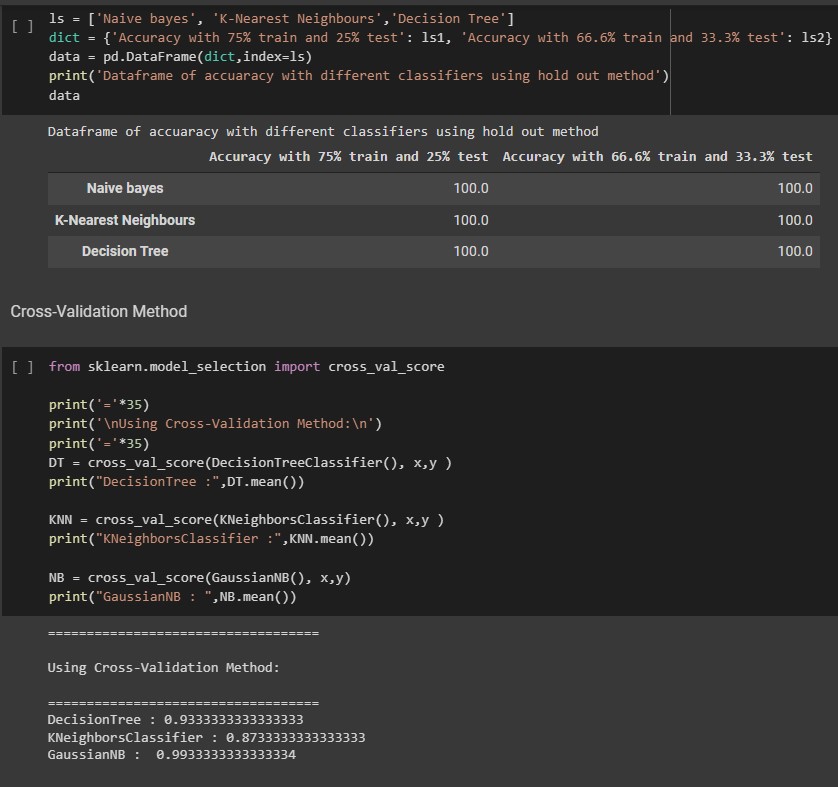


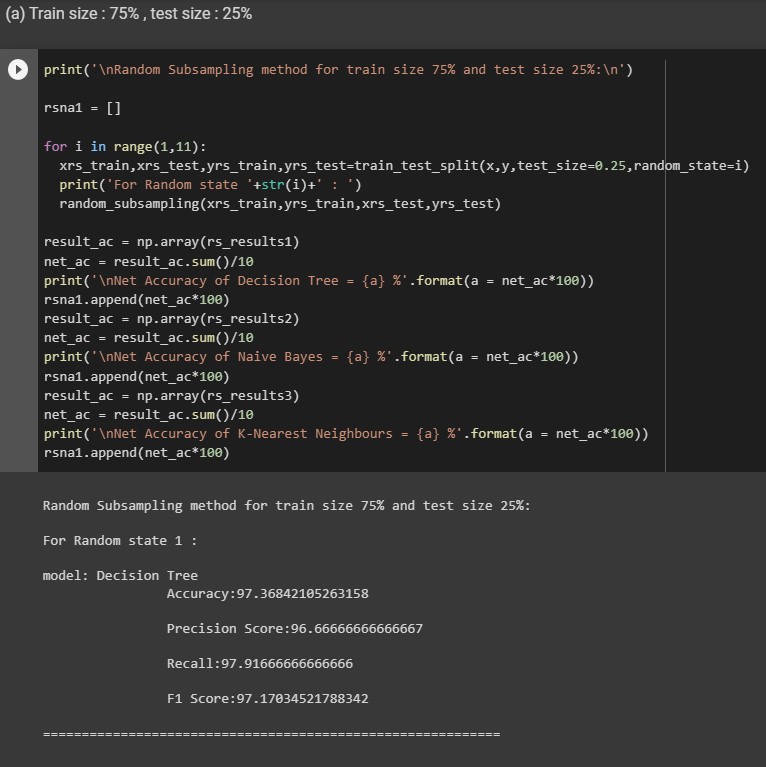


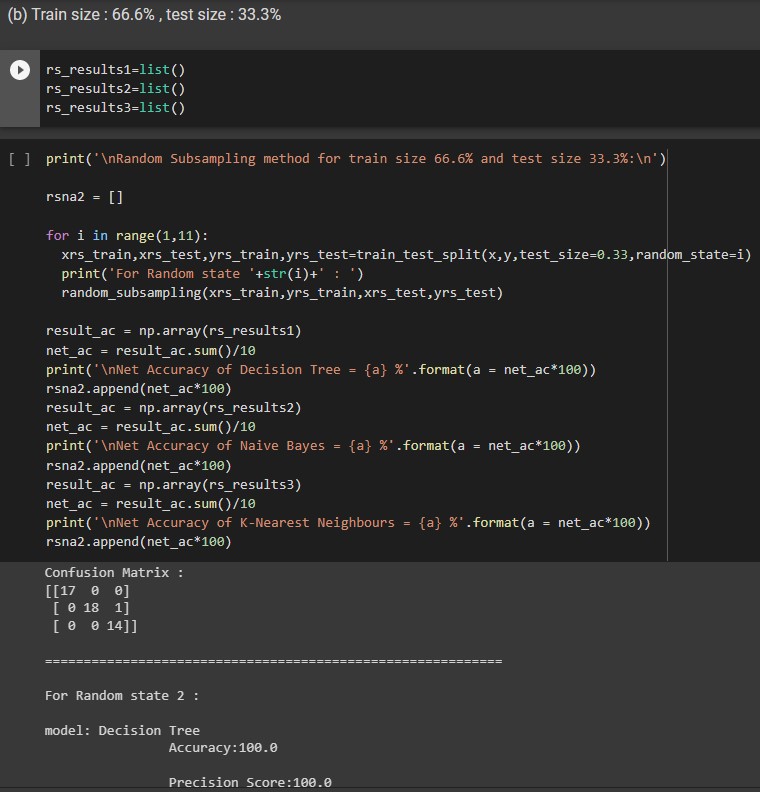


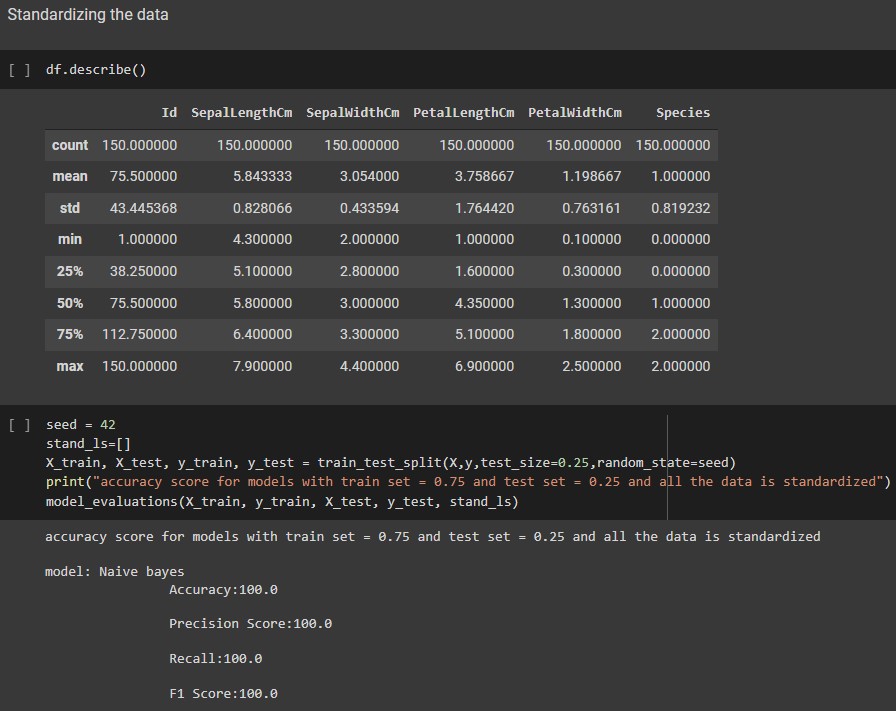


Random Subsampling









|  |
| --- |
| accuracy score for models with train set = 0.75 and test set = 0.25 and all the data is standardized    model: Naive bayes  Accuracy:100.0    Precision Score:100.0    Recall:100.0    F1 Score:100.0    ===========================================================    precision recall f1-score support   1. 1.00 1.00 1.00 15 2. 1.00 1.00 1.00 11 2 1.00 1.00 1.00 12     accuracy 1.00 38 macro avg 1.00 1.00 1.00 38 weighted avg 1.00 1.00 1.00 38      =========================================================== |

model: K

-

Nearest Neighbors

Accuracy:1.0

Precision Score:100.0

Recall:100.0

F1 Score:100.0

===========================================================

precision recall f1

-

score support

15

1.00 1.00 1.00

0

11

1.00 1.00 1.00

1

2

1.00 1.00 1.00

12

accuracy 1.00 38

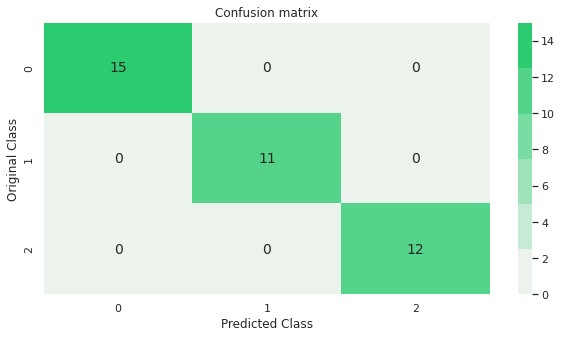
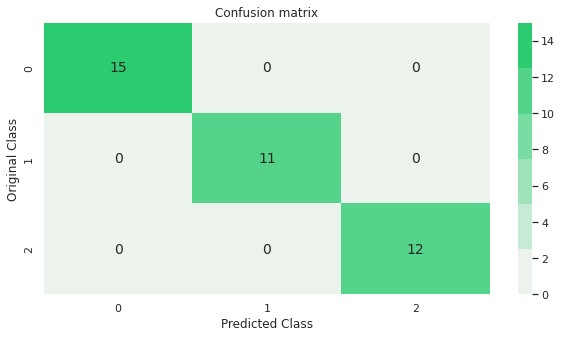
macro avg 1.00 1.00 1.00 38

weighted avg 1.00 1.00 1.00 38

===========================================================

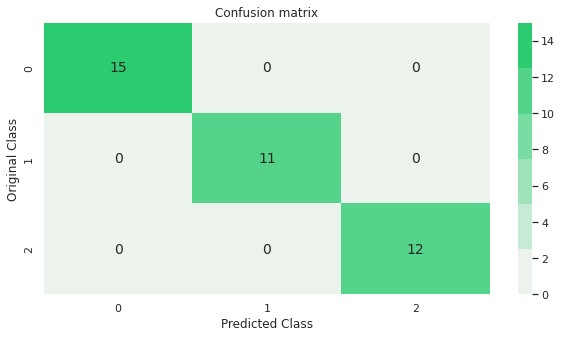
model:Decision Tree

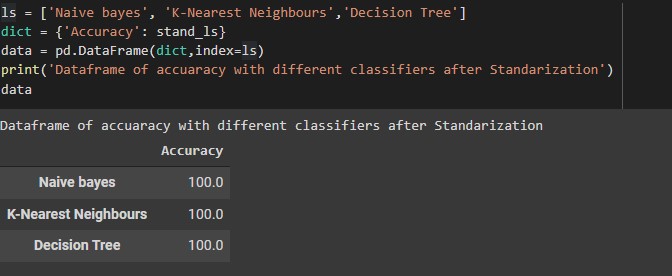
Accuracy:1.0



|  |
| --- |
|  |

|  |  |
| --- | --- |
| Precision Score:100.0    Recall:100.0    F1 Score:100.0    ============================================================    precision recall f1-score support   1. 1.00 1.00 1.00 15 2. 1.00 1.00 1.00 11 2 1.00 1.00 1.00 12     accuracy 1.00 38 macro avg 1.00 1.00 1.00 38 weighted avg 1.00 1.00 1.00 38      ============================================================ | |
|  |  |





Q6. Use Simple Kmeans, DBScan, Hierachical clustering algorithms for clustering. Compare the performance of clusters by changing the parameters involved in the algorithms.

